**Guía para la realización de análisis de enriquecimiento funcional en R (parte dos).**

**Simposio anual Ómicas - Modalidad talleres**

1. **Análisis de enriquecimiento funcional para varias listas de genes (Instalación y carga de archivos)**

En esta sesión se realizará análisis de enriquecimiento funcional para una única lista de genes usando R.

* El archivo para usar es el mismo que el que se ha usado en la sesión anterior. Este se encuentra disponible en el siguiente enlace: https://raw.githubusercontent.com/ccsosa/TALLER\_OMICAS/master/Hallmarks\_of\_Cancer\_AT.csv sin embargo, se llamará directamente desde R.
* Para esta sesión necesitará instalar las siguientes librerías
  1. **gprofiler2**
  2. **biomaRt**
  3. **topGO**
  4. **clusterProfiler**
  5. **GOsummaries**
  6. **Enrichplot**
  7. **org.Hs.eg.db**
  8. **GOCompare**
  9. **ggplot2**

if (!requireNamespace("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("topGO")

BiocManager::install("clusterProfiler")

BiocManager::install("biomaRt")

BiocManager::install("org.Hs.eg.db")

BiocManager::install("enrichplot")

install.packages("gprofiler2")

install.packages("ggnewscale")

install.packages(“devtools”)

library(devtools)

remotes::install\_github("ccsosa/GOCompare")

* Para este ejercicio, el script con las instrucciones se puede descargar del enlace: <https://raw.githubusercontent.com/ccsosa/TALLER_OMICAS/master/Taller_8_GO2.R>

y puede descargarse localmente dando clic derecho y luego usando la opción Guardar como. Posteriormente guarde el archivo como "Taller\_8\_GO\_2.R" entre comillas.

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Al abrir el archivo con extensión .R en RStudio, este será abierto en el área del editor, como se ve a continuación:

Captura de pantalla de computadora

Descripción generada automáticamente

* Para correr los análisis propuestos, primero se deben correr las líneas 3 y 4, en las cuales se cargarán las librerías gprofiler2, biomaRt, topGO, clusterProfiler y GOsummaries.

library (gprofiler2);library(biomaRt);library(topGO),

library (clusterProfiler); library (GOsummaries) ;library(GOCompare)

* Posteriormente se cargará el archivo con la lista de genes pertenecientes a diez categorías (*cancer hallmark)s* en humano que se encuentra localizado en GitHub y se introducen nombres cortos que se usaran más adelante (líneas 10 a la 16)

url\_file = "https://raw.githubusercontent.com/ccsosa/TALLER\_OMICAS/master/Hallmarks\_of\_Cancer\_AT.csv"

x\_group <- read.csv(url\_file)

#leyendo archivo

x\_group[,1] <- NULL

#definiendo categorias

CH <- c("AID","AIM","DCE","ERI","EGS","GIM","IA","RCD","SPS","TPI")

1. **Análisis de enriquecimiento funcional para más de lista de genes**
2. **gprofiler2**

* Ya que se requiere calcular análisis de enriquecimiento funcional para diez listas, procederemos a organizar los datos ya que originalmente estos estan en una matriz de diez columnas como se ve a continuación:

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza media

* El paquete gprofiler requiere que los genes sean organizados por listas y que se tenga un nombre para cada lista ingresada, para esto correremos las líneas 18 a la 26 para generar el formato necesario para nuestros análisis

#preparacion de los datos

x\_Hsap <- lapply(seq\_len(length(CH)), function(i){

x\_unique <- unique(na.omit(x\_group[,i]))

x\_unique <- x\_unique[which(x\_unique!="")]

x\_unique <- as.list(x\_unique)

return(x\_unique)

})

names(x\_Hsap) <- CH

* Ahora se correrá el análisis de enriquecimiento funcional, para este fin, se usará como organism= “hsapiens”, un punto de corte de 0.05, como método de corrección *false Discovery rate*. Este análisis se guardará en el objeto x\_s\_group y la tabla de resultados estará disponible en el objeto res\_group.

(líneas 28 a la 36)

x\_s\_group <- gprofiler2::gost(query = x\_Hsap,

organism = "hsapiens", ordered\_query = FALSE,

multi\_query = FALSE, significant = TRUE, exclude\_iea = FALSE,

measure\_underrepresentation = FALSE, evcodes = T,

user\_threshold = 0.05, correction\_method = "fdr",

domain\_scope = "annotated", custom\_bg = NULL,

numeric\_ns = "", sources = "GO:BP", as\_short\_link = FALSE)

res\_group <- x\_s\_group$result

* Ahora obtendremos el número de términos GO enriquecidos por cada lista de genes. Al correr la línea 39 se obtendrá que la lista de genes asociados a la resistencia a muerte celular tiene más términos enriquecidos.

#Obtener numero de GO enriquecidos por lista de genes

tapply(res\_group$query,res\_group$query,length)

Texto

Descripción generada automáticamente

* Si quiere saber el top tres de términos GO enriquecidos y una tabla de frecuencias con esta información, corra las líneas 41 a 52.

#Obtener top tres GO por categoria

x\_res <- list()

for(i in 1:length(CH)){

x\_res[[i]] <- res\_group[which(res\_group$query==CH[[i]]),]

x\_res[[i]] <- x\_res[[i]][c(1:3),]

}

x\_res <- do.call(rbind, x\_res)

#obtener la tabla de frecuencias para graficar

x\_res$query <- factor(x\_res$query,levels = CH)

other\_table <- table( x\_res$query,x\_res$term\_name)

* Ahora se generará un gráfico de barras con estos top tres de términos corriendo las líneas 54 a la 72. En este grafico verá que hay varios términos GO compartidos entre *cancer hallmarks.*

par(mar = c(13, 4, 11, 1) + 0.2) #add room for the rotated labels

#par(mar=c(1,1,1,1))

bar <- barplot(other\_table,

main = "",

xlab = "", ylab = "Frecuencia",

col = rainbow(10),

xaxt="n",

axes=T,

# legend.text = rownames(other\_table),

beside = F,

las=2,horiz = F,

space=0,cex.names = 0.8)

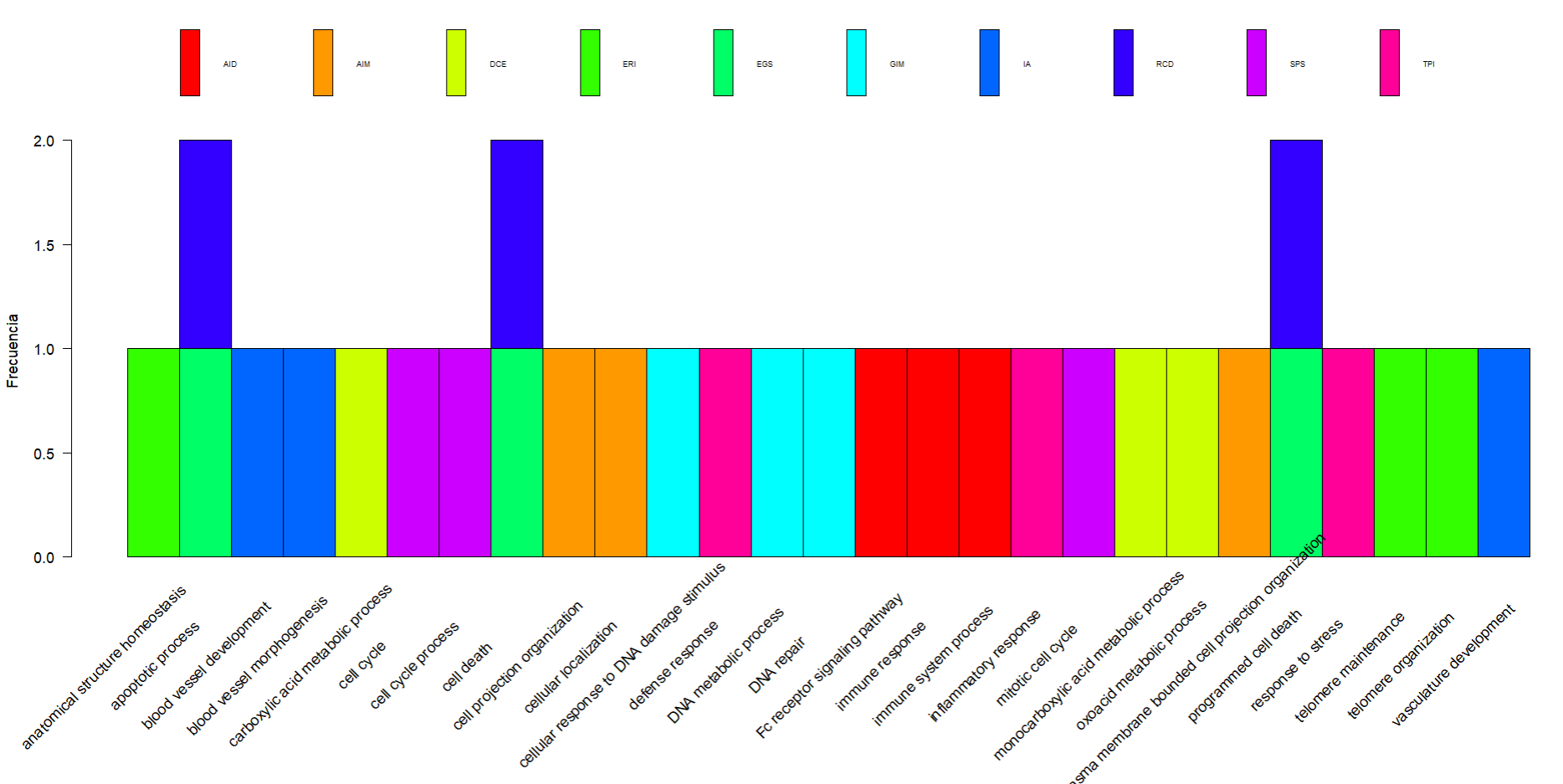
labs <- paste(colnames(other\_table))

text(cex=1, x=bar-1, y=-.52,labs, xpd=TRUE, srt=45)

#axis(2, at = 0:5, labels = 0:5)

legend("top", rownames(other\_table), fill = rainbow(10), bty = "n",horiz = T,inset = c(0,-0.5),xpd = T,

cex = 0.5)



1. **clusterprofiler**

* En esta parte del taller se usará el paquete clusterprofiler, a diferencia de gprofiler los análisis requieren que los ids sean de ENTREZ, es decir que sean de NCBI. Para esto las líneas 79 a 87 permite obtener los ids usando la función bitr.

#Organizar la lista de genes para usarse en clusterProfiler

x\_Hsap\_2 <- list()

for(i in 1:length(x\_Hsap)){

x\_Hsap\_2[[i]] <- clusterProfiler::bitr(as.character(unlist(x\_Hsap[[i]])),

fromType = "SYMBOL",

toType = c("ENTREZID"),

OrgDb = "org.Hs.eg.db")[,2]

}

names(x\_Hsap\_2) <-CH

* Ahora que nuestros genes están con el identificador entrez, se usará la función compareCluster para realizar el enriquecimiento funcional junto con la función enrichGO. Los resultados se guardarán en el objeto clust\_results y el número de términos GO enriquecidos se obtendrá con la función tapply (líneas 90 a 93).

x\_compare <- clusterProfiler::compareCluster(geneClusters=x\_Hsap\_2,enrichGO, OrgDb = org.Hs.eg.db)

clust\_results <- x\_compare@compareClusterResult

tapply(clust\_results$Cluster,clust\_results$Cluster,length)

* Para este caso, la lista de genes asociada a la desregulación de energética celular posee un mayor número de términos enriquecidos

Texto

Descripción generada automáticamente

* Ahora usaremos la función dotplot para visualizar el resultado del análisis de enriquecimiento. En este gráfico, los puntos representan términos GO enriquecidos, el tamaño de los puntos representa la proporción de genes de cada lista y los colores representan los valores p ajustados para cada una de las diez listas de genes. (línea 97)

#Graficar en un dotplot

enrichplot::dotplot(x\_compare)

Imagen que contiene llenado, luz, montón, grupo

Descripción generada automáticamente

* Como alternativa se puede usar la función cnetplot (linea 100), para graficar las interacciones entre genes y términos GO enriquecidos. En este grafo se pueden visualizar el número de genes como tamaño de los nodos, los genes asociados a los términos GO, los cuales son representados como nodos centrales y finalmente cada lista de genes es representada por un color diferente (línea 100).

#Graficar una red

enrichplot::cnetplot(x\_compare)

Varios paraguas de colores

Descripción generada automáticamente

1. **Adaptando resultados de otros paquetes o softwares a clusterprofiler**

* Para este ejercicio usaremos los datos que obtuvimos en el paquete gprofiler, ahora correremos las líneas 106 a 116 para generar una tabla con las columnas necesarias para visualizarse en clusterprofiler

#Modificar archivo de resultados para correr enrichplot

gp\_mod = res\_group[,c("query",

"source",

"term\_id",

"term\_name",

"p\_value",

"query\_size",

"intersection\_size",

"term\_size",

"effective\_domain\_size",

"intersection")]

* A pesar de que los resultados de gprofiler son consistentes necesitamos calcular la frecuencia de genes y la frecuencia de genes con relación a la distribución *background.* También, cambiaremos los nombres de las columnas y los nombres de listas serán convertidas en factores para usarse en los gráficos (líneas 118 a 124).

gp\_mod$GeneRatio = paste0(gp\_mod$intersection\_size, "/", gp\_mod$query\_size)

gp\_mod$BgRatio = paste0(gp\_mod$term\_size, "/", gp\_mod$effective\_domain\_size)

names(gp\_mod) = c("Cluster", "Category", "ID", "Description", "p.adjust",

"query\_size", "Count", "term\_size", "effective\_domain\_size",

"geneID", "GeneRatio", "BgRatio")

gp\_mod$geneID = gsub(",", "/", gp\_mod$geneID)

gp\_mod$Cluster <- factor(gp\_mod$Cluster)

* Ahora se usará la función new para generar un objeto de clase compareClusterResult y otro de clase enrichResult (líneas 126 a 128).

# Convertir de gprofiler a clusterprofiler

gp\_mod\_cluster = new("compareClusterResult", compareClusterResult = gp\_mod)

gp\_mod\_enrich = new("enrichResult", result = gp\_mod)

* Ahora graficaremos el dotplot y el grafo de conectividad como lo hicimos anteriormente (líneas 131 a 135).

#dotplot

enrichplot::dotplot(gp\_mod\_cluster)

#red de interacciones

p2 <- enrichplot::cnetplot(gp\_mod\_cluster)

p2

* Los gráficos se verán como se muestra a continuación, como podrá observar, los resultados entre aproximaciones pueden variar.

Tabla, Calendario

Descripción generada automáticamente Gráfico

Descripción generada automáticamente

1. **GOsumaries**

* Una última opción para análisis de enriquecimiento funcional es presentada a través del paquete GOSummaries. Para esta opción usaremos los datos originales que se cargaron en el CSV, pero los adaptaremos a un formato que lea el paquete GOSummaries (líneas 142 a 147).

#Ajustar el formato a GOSummaries

x\_Hsap3 <- lapply(seq\_len(length(CH)), function(i){

x\_unique <- as.character(x\_Hsap[[i]])

return(x\_unique)

})

names(x\_Hsap3) <- CH

* Ahora graficaremos los resultados y se harán en dos grupos de cinco categorías (líneas 149 a 152). Los gráficos están disponibles en las siguientes dos páginas.

#Correr el analisis y graficar

gs1 = gosummaries(x\_Hsap3)

plot(gs1[1:5])

plot(gs1[6:10])

**Nota**: Si desea ver los resultados, use la siguiente línea de código, ya que son diez listas de genes reemplace el número uno por el de la lista que requiera.

gs1[[1]]$WCD

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. **Comparación de listas de genes para una especie y entre dos especies usando GOCOmpare**

Para esta parte final del taller usaremos el paquete GOCompare, el cual tiene como finalidad permitir la comparación entre listas de genes entre una especie o dos especies a través de la implementación de grafos no dirigidos con pesos para informar acerca de los procesos biológicos más relevantes, proveer un marco de análisis usando distancias de Jaccard y por último usar pruebas de hipótesis para las comparaciones. Actualmente el paquete cuenta con cinco funciones:

|  |  |  |
| --- | --- | --- |
| **Función** | **Foco** | **Descripción** |
| mostFrequentGOs | Una especie | procesos biológicos más frecuentes por categoría |
| graphGOspecies | Una especie | Conversión una lista procesos biológicos y categorías a grafos |
| compareGOspecies | Dos especies | Comparación de dos listas de procesos biológicos y categorías usando PCoA y distancias de Jaccard |
| evaluateGO\_species | Dos especies | Evaluación de la importancia de un proceso biológico para n categorías usando pruebas Chi cuadrado |
| graph\_two\_Gospecies | Dos especies | Conversión una lista procesos biológicos y categorías en dos especies a grafos |

* Para este ejemplo usaremos las funciones compareGOspecies, graphGOspecies y graph\_two\_Gospecies usando dos datasets de ejemplos: data(H\_sapiens\_compress) y data(A\_thaliana\_compress) los cuales son términos GO enriquecidos para genes asociados a cuatro *categorías* de características asociadas a cáncer (*cancer hallmarks)* en humano y sus respectivos genes ortólogos en *Arabidopsis thaliana.*
* Para iniciar este ejercicio cargaremos los datos de ejemplo usando las líneas 158 a 160.

#Cargar datos de ejemplo (cuatro cancer hallmarks)

data(H\_sapiens\_compress)

data(A\_thaliana\_compress)

* Los datos estan guardados como matrices con la siguiente estructura:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

* Dado que comparemos listas de genes y términos GO enriquecidos, debemos establecer las categorías en la columna feature y definir en el código el nombre de columna que tiene los términos GO, que en este caso es “Functional\_Category”. Para este fin usaremos la línea 163.

#Definir la columna que tiene la info de los GO enriquecidos

GOterm\_field <- "Functional\_Category"

* Ya que tenemos dos especies y dos matrices que tienen la misma estructura, debemos decirle al programa el nombre de las especies (líneas 165 a 167).

#Nombrar las especies

species1 <- "H. sapiens"

species2 <- "A. thaliana"

* Ahora procederemos a comparar las dos matrices de términos GO y las dos especies, es decir cuatro categorías y dos especies (ocho grupos para comparar). La función compareGOspecies permitirá está comparación realizando distancias de Jaccard, las cuales se usarán para un análisis de coordenadas principales (*PCoA)*, la matriz de distancias y permitirá extraer los términos GO compartidos y únicos por cada especie (líneas 169 a 176).

x <- compareGOspecies(df1=H\_sapiens\_compress,

df2=A\_thaliana\_compress,

GOterm\_field=GOterm\_field,

species1=species1,

species2=species2)

* Para visualizar el análisis de coordenadas principales debe usar el código de la línea 176. Para este *PCoA,* cada conjunto de categoría y especie es visto como un individuo. Así, el *PCoA* permite ver que tan cercanos son dos especies con relacion a la presencia o ausencia de un término GO.

#graficar el PCoA

x$graphics

Gráfico, Gráfico de líneas

Descripción generada automáticamente

**Nota:** Si desea ver los términos GO compartidos o únicos use los comandos x$shared\_GO\_list y x$unique\_GO\_list

Texto

Descripción generada automáticamenteAplicación

Descripción generada automáticamente con confianza media

* Si desea ver que tan parecidos son la combinación de categorías y especies use la línea 179. Para este caso verá que la desregulación de energética celular es muy similar entre *H. sapiens* y *A. thaliana.*

#cluster con las distancias

plot(hclust(x$distance,"ward.D"))

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

* Si desea obtener la importancia de cada término GO use la función graphGOspecies. Esta función creará un grafo no dirigido con pesos para este fin. Esta opción posee dos opciones para crear grafos, (i) “Categories”, la cual usará las categorías, en este caso (*cancer hallmarks)* y las aristas representan la interacción entre términos. (ii) “GO”, usará los procesos biológicos como nodos y la pertenencia a las categorías como aristas. Esta función está paralelizada y generará grafos muy densos, por lo tanto, debe ser paciente y usar un número de procesadores (*numCores*) que sea adecuado, a su vez, permite guardar el resultado como un archivo graphML que puede verse en Cytoscape. (líneas 182 a 187).

#Extraer pesos para GO enriquecidos en una sola especie

x\_graph <- graphGOspecies(df=H\_sapiens\_compress,

GOterm\_field=GOterm\_field,

option = "GO",

numCores=2,

saveGraph=FALSE,

outdir = NULL)

* Después de correr la función obtendrá una lista con 1,257 nodos y 433,285 aristas. También notará que los términos GO asociados a estrés y a respuesta a estímulos tienen mayores pesos.

Texto

Descripción generada automáticamente con confianza media

* Ahora para finalizar el taller usaremos la función graph\_two\_GOspecies la cual comparará listas de genes entre dos especies. Esta función posee las mismas opciones que la función graphGOspecies, sin embargo, generará un grafo compuesto de tres subgrafos (uno por especie y uno con los términos compartidos) (líneas 190 a 198).

#Extraer pesos para GO enriquecidos entre dos especies y categorias

x\_graph\_two <- graph\_two\_GOspecies(x=x,

species1=species1,

species2=species2,

GOterm\_field=GOterm\_field,

numCores=2,

saveGraph = FALSE,

option= "GO",

outdir = NULL)

* Después de correr la función obtendrá una lista con 1,948 nodos y 456,674 aristas. También notará que los términos GO asociados procesos catabólicos y transporte tienen mayores pesos.

Texto

Descripción generada automáticamente con confianza media